PATENT
ATTORNEY DOCKET NO.: 1120-CA

APPLICATION FOR UNITED STATES LETTERS PATENT

for

SIGNAL PROCESSING INTEGRATED CIRCUIT

by

Joel W. PAGE
16 Laurel Hill
Austin, Texas 78737 U.S.A.

Trenton John GRALE
P.O. Box 17022
Austin, Texas 78760-7022 U.S.A.

Zhuan YE
1271 Deerfield Parkway, #301
Buffalo Grove, Illinois 60089 U.S.A.

Erng Sing WEE
5630 Reamer Street
Houston, Texas 77016 U.S.A.

Sumant SATHE
2910 Medical Arts Street, #207
Austin, Texas 78705 U.S.A.

Sijian CHEN
9114-B Sedgemoor Trail
Austin, Texas 78748 U.S.A.

DATE OF DEPOSIT: 8 March, 2001

EXPRESS MAIL NO.: EL657735133US

A SIGNAL PROCESSING INTEGRATED CIRCUIT

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is related to U.S. Patent Application Serial No._____, filed concurrently herewith by inventors Joel Page, Wai Liang Lee and Erng Sing Wee, entitled "AN INTEGRATED CIRCUIT ARRANGEMENT FOR MULTIPLE-SENSOR TYPES WITH SELECTABLE FRONT ENDS" (Attorney Docket No. 1077-CS).

[0002] This application is related to U.S. Patent Application Serial No. _____, filed concurrently herewith, by inventor Trenton J. Grale, entitled "LOW-POWER LOW-AREA SHIFT REGISTER" (Attorney Docket No. 1076-CS).

[0003] This application is related to U.S. Patent Application Serial No._____, filed concurrently herewith, by inventors Trenton J. Grale and Sijian Chen, entitled "PROGRAMMABLE TEST MODULATOR FOR SELECTIVELY GENERATING TEST SIGNALS OF DELTA-SIGMA ORDER N" (Attorney Docket No. 1121-CS).

BACKGROUND OF THE INVENTION

Field of the Invention

[0004] The invention is directed to a programmable integrated circuit, and more particularly, to an integrated circuit having a programmable front end.

Description of Related Art

[0005] Systems for conducting seismic exploration are well known in the art. On land, a plurality of transducers are deployed over a region and configured to receive reflections of an acoustic signal from different geophysical layers beneath the surface of the earth. In the ocean, arrays of transducers may be towed behind a boat in a spaced configuration in order to

detect those reflections. In transition regions, between land and ocean, sensors may be positioned underwater at fixed locations. Different types of sensors may be utilized for the different environments in which they may be deployed.

[0006] When utilizing a seismic system, a strong acoustic signal is generated by, for example, setting off an explosion or by utilizing an acoustic signal generator having a relatively high power output. Reflections of the acoustic signals from the geophysical layers are then received at the seismic sensors deployed over a given area and the signals recorded, typically, for later analysis.

[0007] In some configurations, a seismic sensor is co-located with an analog to digital converter, such as a delta-sigma modulator, which converts an analog signal from the sensor into a digital signal for recording and processing. Seismic exploration has exacting requirements for seismic sensors and for the electronics which process the signals derived from those sensors. There is therefore a need to be able to test both the sensors and related equipment to ensure that both devices and the associated electronics are functioning properly.

Summary of the Invention

[0008] The invention is directed to a programmable integrated circuit, and, more particularly, to an integrated circuit having a programmable front end for selectively processing incoming signals having different characteristics.

[0009] Still other objects and advantages of the present invention will become readily apparent to those skilled in the art from the following detailed description, wherein only the preferred embodiment of the invention is shown and described, simply by way of illustration of the best mode contemplated of carrying out the invention. As will be realized, the invention is capable of other and different embodiments, and its several details are capable of

modifications in various obvious respects, all without departing from the invention. Accordingly, the drawing and description are to be regarded as illustrative in nature, and not as restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Figure 1 is a block diagram of an integrated circuit implementing various aspects of the invention.

[0011] Figure 2 is a block diagram showing various sensor modulator combinations which can be selectively utilized with the integrated circuit of Figure 1.

[0012] Figure 3 is a block diagram showing a plurality of sets of power-ground combinations found on the integrated circuit illustrated in Figure 1.

[0013] Figure 4 is a block diagram showing the interface circuitry used to selectively interface different sensor/modulator, combinations to a decimation chain.

[0014] Figure 5 is a block diagram showing the interface of Figure 4 in more detail.

[0015] Figure 6 shows the sinc decimation chain including selectable front ends.

[0016] Figure 7 shows coefficients and the mathematical representation of the Sinc1A filter implementation.

[0017] Figure 8 is a block diagram of the Sinc1A implementation in accordance with the invention.

[0018] Figure 9 is depicts the coefficients and the mathematical representation of the Sinc1B filter.

[0019] Figure 10 is a block diagram showing an exemplary implementation of the Sinc1B filter in accordance with the invention.

[0020] Figure 11 is a table showing the mathematical representation and the filter coefficients for the Sinc2(a), Sinc2(b), Sinc2(c), Sinc2(d) and Sinc2(e) filters.

[0021] Figure 12 is a representation of the Sinc 2(a) and 2(b) filters, showing the shifts and adds utilized to implement the filters.

[0022] Figures 13A, 13B and 13C show the shift add implementation of Sinc filters Sinc2(c), Sinc2(d) and Sinc2(e), respectively.

[0023] Figures 14A and 14B show implementation of the Sinc2(a) and Sinc2(b) filters with their respective common accumulate phase and output phase.

[0024] Figures 15A and 15B show implementation of the Sinc2(d) filter with its respective accumulate phase and output phase.

[0025] Figures 16A, 16B and 16C show implementation of the Sinc2(c) filter with its accumulate phase A, accumulate phase B and output phase representations.

[0026] Figures 17A and 17B illustrate the implementation of the Sinc2(e) filter in an accumulate phase and an output phase, respectively.

[0027] Figures 18A though 18I show implementation of the Sinc2 filter with the accumulate-output architecture with respective time multiplexing phases.

[0028] Figures 19A and 19B are memory diagrams showing the relationship between the main program and the subroutines used to implement the Sinc filters.

[0029] Figure 20 is a block diagram showing an architecture for carrying out the adds and shifts needed to implement the Sinc2 filter.

[0030] Figure 21 is a flow chart of a programming procedure for the Sinc filters.

[0031] Figure 22 is a flow chart of a process for determining coefficients to prevent overflow in internal registers when implementing the Sinc filters.

[0032] Figure 23 is a block diagram of a DSP utilized for filtering and decimation in accordance with the invention.

[0033] Figures 24A and 24B show the address segmentation of the program and data address space for the DSP.

[0034] Figure 25 shows the implementation of the execution unit shown in Figure 23.

[0035] Figure 26 shows the data address unit shown in Figure 23 in more detail.

[0036] Figure 27 shows the program address unit of Figure 23 in more detail.

[0037] Figure 28 shows the structure of an SDOR output data word and mapping of the internal registers to the output word.

[0038] Figure 29 shows connection of a plurality of the integrated circuits shown in Figure 1 to operate in a daisy chained serial data token arrangement.

[0039] Figure 30A is a block diagram showing the TMOD buffers and filters shown in Figure 1 in more detail.

[0040] Figure 30B is a block diagram showing a mathematical representation of a digital Σ modulator 3030A or 3030B of Figure 30A.

[0041] Figures 30C1 through 30C10 identify symbols used in Figures 30D through 30H.

[0042] Figure 30D shows one implementation of the digital Σ modulator of Figure 30B using a fully parallel one clock system with multipliers.

[0043] Figure 30E shows one implementation of the digital Σ modulator of Figure 30B using shifts and adds instead of multipliers.

[0044] Figure 30F1 shows one implementation of the digital Σ modulator of Figure 30B using a pipeline to perform feedward summing and integration.

[0045] Figure 30F2 is a system state table for the implementation shown in Figure 30F1.

[0046] Figure 30G1 shows one implementation of the digital Σ modulator of Figure 30B using a hybrid memory system.

[0047] Figure 30G2 is a system state table for the implementation shown in Figure 30G2.

[0048] Figure 30H1 shows one implementation of the digital Σ modulator of Figure 30B that attempts to reuse hardware as much as possible.

[0049] Figure 30H2 is a system state table for the implementation shown in Figure 30H1.

[0050] Figure 31 shows the micro control path for operation of the test signal modulator.

[0051] Figure 32 shows the data path portion of the TMOD micro control path shown in Figure 31.

[0052] Figure 33 shows exemplary steps needed to implement an exemplary algorithm for the test signal and modulator shown in Figure 31.

[0053] Figure 34 shows the actual programming loaded into microinstruction register of Figure 31 to implement the algorithm shown in Figure 33.

[0054] Figures 35A and 35B show an improved shift register with reduced area and power and a table representing the semantics of various signal lines for that shift register, respectively.

[0055] Figure 36 shows a plurality of time lines showing an exemplary clock alignment associated with on-chip generation of clocks as shown for example in item 140 of Figure 1.

[0056] Figure 37 is a flow chart of a process for programming clocks in accordance with the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0057] Figure 1 is a block diagram of an integrated circuit implementing various aspects of the invention. Integrated circuit 100 represents a lower power programmable decimation filter in accordance with the invention. It utilizes a low power general purpose digital signal processor (DSP) 120 to achieve efficient filtering for up to four delta-sigma modulators. It is used with executable code, described hereinafter, to perform many different filter functions. These include linear and minimum phase filters at a variety of output word rates. It can service between one and four channels of incoming data, on a user selectable basis, received over modulator data interface 110. The filter decimated output is applied to serial data output register 130. A bus 115 interconnects the modulator data interface 110, programmable decimation and filtering engine 120 and serial data output register 130. In

addition, the bus provides access to a clock and sync generation module 140, two test modulator buffer and filters 150 and 160, a serial control interface 170, a serial peripheral interface 180, a general purpose I/O interface 190 and a watch dog timer 192. A JTAG interface 191 is provided for testing, such as boundary scan, on the integrated circuit.

[0058] Figure 2 is a block diagram showing various sensor modulator combinations which can be selectively utilized with the integrated circuit of Figure 1. A modulator data interface 110 of Figure 2 is designed to accommodate a plurality of different types of sensor/modulators. Exemplary sensor modulator packages might include the CS5321 and the CS5372 chips provided by Crystal Semiconductor, Inc. of Austin, Texas. Additional sensor outputs might also be accommodated such as shown at 200C. Integrated circuit 100 is programmable to accommodate a variety of input rates and algorithms that might be utilized by various input devices.

[0059] Figure 3 is a block diagram showing a plurality of sets of power-ground systems found on the integrated circuit illustrated in Figure 1. Integrated circuit 100 utilizes three separate power supply-ground systems. Preferably it is selectable based on a needed or desired operating frequency. In this particular exemplary embodiment, an external 5 volt power supply is applied over pins 11/25 of the chip and serves as a power source VDDPAD2. The ground or return pins for that supply are pins 24 and 38 of the exemplary implementation chip 100. A second supply is, in this example, a 3.3 volts power pin which appears at 54 on the integrated circuit 100. The return for this 3.3 volts pin is found on pin 53. The second supply system services the output side whereas the first system service the input side. A third independent supply is found on pins 7 and 40. This one is typically 2.5 volts and supplies the digital processing circuitry on the chip. The return for this third supply is found on pins 6 and 39 of the integrated circuit 100. In the arrangement shown, the substrate of the digital processing is connected to the ground pin 24/38 of the integrated circuit chip. There are thus, three

independent power supply-ground systems utilized on the chip which permits a variety of isolation and power needs to be accommodated.

[0060] Figure 4 is a block diagram showing the interface circuitry 110 used to selectively interface different sensor/modulator, combinations to a decimation chain (Sinc1 (111) and Sinc2 (112)). The input lines MDATA connect to the modulator interface 110 prior to being applied to the programmable decimation and filtering engine 120 where a decimation chain is programmably implemented.

[0061] Figure 5 is a block diagram showing the interface 110 of Figure 4 in more detail. The input from the MDATA and MFLAG pins on integrated circuit 100 are applied in parallel, directly or indirectly to three different latches, 500A, 500B and 500C. In the exemplary embodiment shown, latch 500A is set up to handle 128 Kbps input rates, latch 500B is set to handle 512 or 256 Kbps, selectable, inputs and latch 500C is set up to handle 256 Kbps or 128 Kbps inputs. Latch 500A might be utilizable with a micromachine input data stream whereas latches 500B and 500C are utilizable with the Crystal semiconductor products previously discussed. A selector 510 is controlled by an input signal mdi_cfg which selects the output to be utilized for the remainder of the circuitry. The output of the selector 510 is applied to selectors 520A and 520B. Selectors 520A and 520B permit the incoming data stream to be selectively routed to the Sinc1a filter input circuitry 530A or the Sinc1b circuitry 530B. Selectors 520A and 520B permit the outputs of TMOD buffer and filter 1 (150) and TMOD buffer and filter 2 (160) to be applied, selectively, to the inputs of the Sinc1a interface 530A and the Sinc1b interface 530B, respectively.

[0062] Figure 6 shows the Sinc decimation chain 120 including selectable front ends. The outputs from the Sinc1a interface 530A and the Sinc1b interface 530B from Figure 5 are applied to Sinc1 filters which can be selectively activated. The Sinc 1a and Sinc 1b filters are preferably implemented in hardware. The Sinc1a filter 600A is a 5th order decimate by 8

Sinc filter. Sinc filter 1b (600B) is a decimate by 2, 6th order Sinc filter. The output of the Sinc1a filter and the Sinc1b filter are applied to selector 610 which produces a 64 kHz output at 17 bits wide. This output serves as the input to the Sinc2 filter chain which is comprised of a plurality of Sinc filters, preferably implemented in software including, in this example, Sinc2a (620), Sinc2b (630), Sinc2c (640), Sinc2d (650) and Sinc2e (660). The arrows depicted in the Sinc2 area of Figure 6 represent different paths that can be selectively followed by the data, in accordance with programming direction, to implement a programmable Sinc filter. Thus, Sinc2 is a programmable Sinc filter with variable decimation ratios ranging from 2 through 16.

[0063] Figure 7 indicates the mathematical representation of a 5th order decimate by 8 Sinc filter. Coefficients for one half of the symmetrical set are also given.

[0064] Figure 8 is a block diagram of the Sinc1a implementation in accordance with the invention. The incoming data bits from a data stream are represented as if they were loaded from head to tail in a register 890. The use of a register as a memory element represents a convenient representation but other forms of memory may be used. The head 2-bits and the tail 2-bits are read into ROM 800. ROM 1 (810) and ROM2 (820) receive byte-0 and byte-1, respectively, from the data stream. These bytes are summed in adder 830 and then summed with the output of the head tail ROM in adder 840 to create an interim sum at register 850. During a second phase, ROM 1 receives byte-3 and ROM 2 receives byte-2 from the data stream in twisted (bit reversed) order. The output of ROM 1 and ROM 2 is then summed at 830 and summed with the output of the register 850 from the previous stage to produce a final answer.

[0065] This architecture is based on using look up tables instead of performing multiplications. The coefficients are symmetric and this permits one to use one half the expected number of look up tables.

[0066] The Sinc1b filter is represented mathematically as shown in Figure 9. The impulse response of this filter is shown mathematically in this Figure as well.

[0067] Figure 10 is a block diagram showing an exemplary implementation of the Sinc1b filter in accordance with the invention. Data from one of the incoming channels is selected using multiplexor 1000 based on 2-bits of signal channelData_mux. During a first phase (Phase 0) the first 3-bits of the channel word are applied through selector 1010 to the address input of a 3-bit ROM 1020. The middle bit from the word is applied to middle bit converter 1030 which generates a constant value. The output of the middle bit converter and the output of the 3-bit addressed ROM is applied to adder 1050 and the results stored in register 1060 on the positive edge of a 512 CLK. During the second phase (Phase 1) the last 3 bits are twisted (i.e. applied in bit reversed order) and used as an index into the ROM 1020. The sum of the output of the addressed ROM 1020 is added to the contents of temporary register 1060 and the result applied to an accumulate register 1070 for output to the selected data channel using multiplex 1080.

[0068] Figure 11 is a table showing the mathematical representation and the filter taps for the Sinc2(a), Sinc2(b), Sinc2(c), Sinc2(d) and Sinc2(e) filters.

[0069] Figure 12 is a representation of the Sinc2(a) and 2(b) filters, showing the shifts and adds utilized to implement the filters.

[0070] Figures 13A, 13B and 13C show the shift add implementation of Sinc filters Sinc2(c), Sinc2(d) and Sinc2(e), respectively.

[0071] Figures 14A and 14B show the accumulate phase and the output phase, respectively which can be utilized to implement the filter of Sinc2a and Sinc2b.

[0072] Figures 15A and 15B show the accumulate phase and the output phase which can be utilized to implement the Sinc filter Sinc2d.

[0073] Figures 16A, 16B and 16C show implementation of the Sinc2(c) filter respectively in an accumulate phase A, accumulate phase B and output phase representations. Figures 16A and 16B represent two accumulate phases for implementing the Sinc2c filter. Figure 16C represents an output phase. These two accumulate in phases and one output phase can be utilized to implement the Sinc2c filter.

[0074] Figures 17A and 17B illustrate the implementation of the Sinc2(e) filter in an accumulate phase and an output phase, respectively. Figure 17A illustrates the accumulate phase and Figure 17B illustrates the output phase of an implementation which can be utilized to implement the Sinc2e filter.

[0075] Once each of the many Sinc filters have been arranged for shifts and adds and organized into accumulator and output subroutines, one needs to arrange the order of the execution of the subroutines to accommodate all the calculations. The number of additions/subtractions one can perform is based on the spacing between input values received from Sinc1. Because Sinc2 runs at 512 kHz, for an input from Sinc1 of 64 kHz, we have the ability to do eight addition/subtractions per input word.

[0076] Figures 18A though 18I show implementation of the Sinc2 filter with the accumulate-output architecture used with time multiplexing. This shows an example of the spacing and time allocation which can be utilized to perform the accumulate outputs steps needed for implementing the Sinc2 filters on an ongoing basis. The example implements a decimate by 16 in Sinc2. A different arrangement would be required for different decimation rates in Sinc2. Exemplary code for implementing this is shown in an Appendix.

[0077] Figures 19A and 19B are memory diagrams showing the relationship between the main program and the subroutines used to implement the Sinc filters. The programming structure of the Sinc2 filters utilizes a plurality of subroutines called from a main program. In Figure 19 the

main program is shown implemented in RAM 1, the subroutines are shown implemented in RAM 2. Each statement in the main program will call the subroutines of RAM 2 for execution.

[0078] Figure 20A shows a control-datapath architecture for implementing the Sinc2 filters. Sequence control 1 (2001A) involves a read state or RAM1 at a specified address. The contents of RAM1 points to an entry point for a subroutine located in RAM2. Sequence control 2 involves a read beginning at that entry point and sequences through the instructions of the subroutine and outputting those instructions as datapath control signals. The datapath itself is shown in Figure 20B.

[0079] Figure 20B describes an architecture for implementing the Sinc2 channel Datapath. Data is read into a plurality of registers 2000i where i represents the number of the register. The contents of a particular register can be selected by selector 2010 and applied to shifter 2020 which can shift at left or right as needed. The output of shifter 2020 is applied to add or subtractor 2030, the output of which is fed both the input of the register set 2000i by a selector 2040 or to the input of selector 2050 for application to a register of register set 2060i. Contents of a particular register can be selected using selector 2070 and applied to shifter 2080 where it can be selectively shifted and then the output of 2080 is applied to add or subtractor 2030. Where it can be selectively added to the contents output from shifter 2020.

[0080] The steps which can be utilized to create a program for operating the Sinc filters will be described with respect to an example.

[0081] **Step 1** -- Chose the desired decimation rate. In this example, we will utilize 16.

[0082] **Step 2** -- Select which filters need to be involved in the decimation. This can be done conveniently by reference to Figure 6, where one can readily see that mini-sinc filters 2a, 2b, 2d and 2e can be utilized to achieve a decimation ratio of 16.

[0083] **Step 3** -- Separate coefficients into form suitable for shift-add operations. This can be done from the mathematical representation using an approach similar to that shown in Figure 12.

[0084] **Step 4** -- Check for overflow after each addition in the filter. See the discussion of Figure 22 which follows.

[0085] **Step 5** -- Perform the necessary truncation to 24-bits and scaling of subsequent coefficients in mini-sincs. See the discussion of Figure 22.

[0086] **Step 6** -- Time multiplex accumulate in output subroutines so that a maximum of eight operations can occur from each input from Sinc1.

[0087] **Step 7** -- Create code for RAM2 (Accumulate and Output Subroutines) in the form: [Coeff 1] [Src 1] [Src 2] [Dest] [Coeff2] [Done Subroutine].

[0088] **Step 8** -- Create code for RAM1 (Main Control code): [Line #] [Wait for new data] [Done program].

[0089] Figure 21 is a flow chart reflecting these steps.

[0090] Figure 22 is a flow chart of a process for determining coefficients to prevent overflow in internal registers when implementing the Sinc filters. This flow chart describes a process by which one can check filter coefficients to overflow in internal registers. From a given 1's density, one determines the maximum value entering Sinc2 from Sinc1 (Sinc1a or Sinc1b) (2200). One then propagates this maximum value through the direct-transposed form of each of the mini-sinc filters making up the filter chain to be programmed in Sinc2 (2210). After each adder is encountered, one checks to see if saturation has occurred. If saturation has occurred (2220-yes) to prevent overflow, one shifts the operands of the saturation in addition to the right before addition. Specifically, this implies shifting the result of the previous storage register to the right (extending as needed). One also needs to shift all

subsequent coefficients in the current mini-sinc to the right by the same number of shifts so that the addition "Input * Coefficient +previous storage register" works properly. Once that is done, one returns to step 2210.

[0091] If saturation does not occur (2220-no), one checks (2230) to see if the complete Sinc filter processing has been completed, if not, one returns to step 2210 for further processing. If processing has been completed (2230-yes), the process ends (2250).

[0092] Figure 23 is a block diagram of a DSP utilized for filtering and decimation in accordance with the invention. The DSP utilized as the filtering and decimation engine 120 of Figure 1 has an execution unit 2300 which services two buses, an SRCA bus 2320 and an SRCB bus 2310. The SRCB bus has a program address unit 2340 and boot ROM 2345B and program and coefficient RAM 2345A, a data address unit 2330 is connected both to bus 2310 and to bus 2320. Data RAM 2335A and Data ROM 2335B are also connected to SRCA bus 2320. Data comes in and out over a bus 2320 by DSP I/O registers (shown in dashed). The execution unit 2300 has access to accumulators 2350 and 2360. These may be independently accessed over the bus 2320. A decoder 2370 interfaces the SRCB bus 2310 with an I/O control bus.

[0093] Figures 24A and 24B show the address segmentation of the program and data address spaces for the DSP. The data memory space may be partitioned between RAM and ROM as shown or may be implemented in RAM alone, without any ROM.

[0094] Figure 25 shows the implementation of the execution unit shown in Figure 23. The execution unit 2300 is shown more in detail in this Figure. Data is received from the SRCA and SRCB buses via 24-bit latches 2540 and 2530, respectively. An arithmetic logic unit 2500 can receive the output from the latches. A 24 x 24 multiplier 2510 and a 54-bit adder 2520 complete the execution unit.

[0095] Figure 26 shows the data address unit shown in Figure 23 in more detail. The data address unit contains three register sets 2600i (AR0-AR7), 2610i (IDAR0-IDAR7) and 2620i (MAR0-MAR7). An instruction word received from bus SRCB can independently specify both the A operand and the destination. The A operand can be the contents of an AR, IDAR, MAR or I/O register. It can also be a location in data memory. The register sets are available for direct addressing by the DSP. When A operand is a location in data memory, the instruction word can specify the 7 least significant bits of the data memory address (direct addressing) or an address reference which contains a data memory address (register indirect addressing).

[0096] Figure 27 shows the program address unit of Figure 23 in more detail. The program address unit consists of two 14-bit Program Address Registers (PAR) 2700A and 2700B, two 14-bit Modulo Program Address Registers (MPAR) 2710A and 2710B, the 14-bit Program Counter (PC) 2720 a 14-bit Loop Counter 0(LC) 2730 and 15 stack locations each for the PC and LC. There is also a stack pointer which points to the current PC and current LC.

[0097] Figure 28 shows the structure of a Serial Data Output Register 130 (SDOR) output data word and mapping of the internal registers to the output word. The serial data output register 130 of Figure 1 consists of two 24-bit internal registers, SD_STAT containing status and control bits, and SD_DATA, containing the data. Internally, these are at least 4-deep FIFOs to store up to four data words to be output. Preferably, they are 8 deep. These internal registers are mapped to the output word as shown in Figure 28. The word is essentially divided into three parts, the first is a status component, the second is an overwrite bit and the third is a data word. 2-bits of the status words represent the filter address (0 through 3). 2-bits represent the modulator channel number (channel 1 through channel 4) an additional bit indicates whether a time break will be utilized, 1-bit indicates whether data from the DSP or Sinc modulator has overflowed. The W-bit indicates that

channel data has been overwritten. The W-bit indicates a port overflow condition and is set by hardware when the DSP overwrites a register whose data has not been sent.

[0098] Figure 29 shows connection of a plurality of the integrated circuits shown in Figure 1 to operate in a daisy chained serial data token arrangement. Integrated circuit 100 of Figure 1 can be connected in a bus arrangement shown in Figure 29. It uses a bus token scheme which operates as follows. The filters are daisy chained as illustrated in Figure 29 so that the SDTKO pin on one filter connects to the SDTKI pin on the next filter in the chain. When a filter finishes transmitting all of its data words, it enables the output buffer on SDTKO so that the next filter in the chain can latch the token bit input on the SDTKI pin. If a filter does not have any data to transmit it immediately shifts the token bit to the next node. If the filter receives a bus token while SD_DATA and SD_STAT FIFOs are only partially filled, it immediately passes a token onto the next filter. The microcontroller must initialize the token scheme by driving the SDTKI input pin on the first filter in the daisy chain. The microcontroller also must receive the token bit from the SDTKO output pin of the last filter in the daisy chain and send it back to the first filter.

[0099] Figure 30A is a block diagram showing the test signal modulator (TMOD) buffer and filters shown in Figure 1 in more detail. Each of test modulator buffer and filters 1 and 2 (150 and 160, respectively), are substantially identical from a hardware perspective. The DSP 120 provides data to FIFO 3010A/B which is then utilized to feed the programmable interpolator 3020A/B which then drives a delta-sigma modulator 3030A/B to produce a delta-sigma version of the input signal provided by the DSP. An example would be the DSP providing 24-bit data of a 31.25 Hz sinc wave at 4 kHz to the TMOD. The TMOD would interpolate this delta-sigma $\Delta\Sigma$ modulator to a 1-bit, 256 kHz representation. The DSP can provide a highly accurate sinc wave or wave form of some other shape for testing on the

system. The wave shape provided by the DSP would then be represented in a digital delta-sigma output format for use in testing. The output delay 3040 A/B permits the phase delay of the wave form generated by the DSP to be adjusted with considerable precision.

[00100] The TMOD is designed to perform digital delta-sigma modulation, receiving 24-bit input data and generating 1-bit output data and CLK. It is implemented using a programmable microsequencer. It produces an output bit by executing a sequence of microinstructions. Because of its programming flexibility, it can perform several variations on the basic digital delta-sigma modulator algorithm.

[00101] Figure 30B is a block diagram showing a mathematical representation of a digital Σ modulator 3030A or 3030B of Figure 30A. The output of interpolators 3020A and 3020B are applied to the respective *px input of the corresponding modulator, where it is summed with a voltage Usef, the polarity of which is controlled by feedback input MSB. MSB also is applied to output y[i] as the Σ output (1Bπin this example). Higher frequency components tend to follow a different path than low frequency components.

[00102] Figures 30C1 through 30C10 identify symbols used in Figures 30D through 30H.

[00103] Figure 30D shows one implementation of the digital Σ modulator of Figure 30B using a fully parallel one clock system with multipliers.

[00104] Figure 30E shows one implementation of the digital Σ modulator of Figure 30B using shifts and adds instead of multipliers.

[00105] Figure 30F1 shows one implementation of the digital Σ modulator of Figure 30B using a pipeline to perform feedward summing and integration.

[00106] Figure 30F2 is a system state table for the implementation shown in Figure 30F1.

[00107] Figure 30G1 shows one implementation of the digital Σ modulator of Figure 30B using a hybrid memory system. The circuit uses a RAM and a 2 stage data pipeline to reduce the number of RAM reads/writes.

[00108] Figure 30G2 is a system state table for the implementation shown in Figure 30G2. Since many of the components are reused for similar operations, the control for the hybrid memory system has more states than the other system.

[00109] Figure 30H1 shows one implementation of the digital Σ modulator of Figure 30B that attempts to reuse hardware as much as possible.

[00110] Figure 30H2 is a system state table for the implementation shown in Figure 30H1.

[00111] By nature of their design, these systems can easily be adapted or modified for different configurations. The programmable nature of the integrated circuit described herein permits selective implementation of two or more of these different architectural implementations of a Σ modulator. Not only can the particular implementation, be selected, but the order N of a particular implementation can be selected by selecting different coefficient sets.

[00112] The particular architecture of an implementation, as well as the order N of the algorithm can be set and changed by control signals, for example by those originating from a sequencer.

[00113] Figure 31 shows the micro control path for operation of the test signal modulator. In operation, a user microprogram resides in microinstruction registers P0 through P7 (3100l). Each of the

microinstructions contains bits that control the various compound that is in the data path to perform bus enabling, addition and subtraction, arithmetic bit shifting, and register writing. When the TMOD is running, the microsequencer cycles through the microinstruction registers, enabling each one under the instruction bus in turn. By writing appropriate instructions, the user can cause the TMOD to perform almost any delta-sigma algorithm up to 5th order with one resonator. The bits of the microinstruction word are applied to the data path circuitry, shown in Figure 32, hereinafter, to implement the delta-sigma algorithm defined by the microinstruction programming.

[00114] Figure 32 shows the data path portion of the TMOD micro control path shown in Figure 31. The data path is really divided into two data paths, one to perform the integration and one to form the feed forward summation. The integration data path consists of the integration bus (I bus) in the integration ALU (the I ALU), the integration registers D1 through D5 and other registers that drive the I bus. The summation data path consists of the sum bus (S bus) and the sum ALU (S ALU), the feed forward sum register (SUM) and the sum sign register for the output bit stream. The S ALU can also be used to compute temporary quantities to be stored in temp and used in the integration process.

[00115] An example will now be given of the programming of the TMOD buffer and filter device in carrying an exemplary algorithm.

[00116] Figure 33 shows exemplary steps needed to implement an exemplary algorithm for the test signal and modulator shown in Figure 31. Figure 33 shows the steps needed to implement a particular delta-sigma modulation algorithm. The functional steps that are needed to implement that algorithm correspond to various combinations of actions on the control and signal lines of the data path shown in Figure 32.

[00117] The TMOD architecture consists of data and control registers, arithmetic logic units and buses. Most of the registers are internal and are

accessible to the DSP indirectly. Internal registers include microprogram registers; feedback constant registers; configuration bits for interpolation factor, CLK rate, output and output delay, and data registers for integration in a feed forward sum. The DSP interfaces to the TMOD through two DSP I/O registers, TMODCFG and TMODDAT. The DSP uses TMODCFG and TMODDAT to configure the TMOD and uses TMODDAT to supply data during operation. During programming, the DSP writes control bits in TMODCFG which causes the contents of TMODDAT and some bits of TMODCFG to be strobed into a selected internal register. When the TMOD is running the DSP supplies the input data by writing to TMODDAT.

[00118] Figure 34 shows the actual programming loaded into microinstruction register shown in Figure 31 to implement the algorithm shown in Figure 33. Given the steps shown in the algorithm shown in Figure 33, one can determine the states of the various bits needed in order to place the hardware shown in Figure 32 into the states necessary to implement the algorithm.

[00119] An example of the translation from the algorithm shown in Figure 33 to the programming required to implement that algorithm in the hardware shown in Figure 32 is shown in Figure 34. Thus, the binary programming needed for the microinstruction registers 3100I of Figure 31 is that shown in Figure 34. By loading these instructions from Figure 34 into the instruction registers of 3100I, of Figure 31, one can implement the algorithm shown in Figure 33.

[00120] Figures 35A and 35B show an improved shift register with reduced area and power and a table representing the semantics of various signal lines for that shift register, respectively. The output delay shown in Figure 30 at 3040 A/B is preferably implemented as shown in Figure 35A. The idea is to segment the shift register into smaller shift registers. This breaks up the multiplexing that must be done at either the input or the output, and thus reduces logic area for bit selection. Power is also conserved because

segments that are not used see no data transitions. It would also be possible to switch off the CLK to unused segments so as to reduce power consumption further or to eliminate power to unused segments entirely.

[00121] The architecture described results in a much simpler implementation from that which would be required to achieve a commensurate delay flexibility in the prior art.

[00122] The register illustrated achieves a delay of from 0- to 63-bits selectively. It does so in this manner by having a 15-bit register D with 16 individual taps shown which will permit delays from 0- to 15-bits.

[00123] If more than 16-bits delay is required, one or more of the 16-bit untapped shift registers A through C are selected for receiving the undelayed input. Thus, the delays are divided into four segments. Three of the segments have fixed delays of 16-bits. And one segment has a selectable delay from 0- to 15-bits. The amount of the delay is specified by a 6-bit word. The bits of that word are described in Figure 35B. The undelayed incoming data is applied to a pin that is unique to each data segment of a selector. The particular segment to receive the undelayed data is selected by bits 4 and 5 of the delay word. If the bits are 00, then the undelayed data is applied to segment D only. If the bits are 01, the undelayed data is applied to segment C and followed by being applied to segment D with an output tab specified by bits 0 through 3 of the delay word. In this manner, one can achieve 0-63-bits delay using only 16 taps and in a way that permits power to be reduced considerably.

[00124] The maximum number of bits of delay in a segment and the maximum number of segments to be used in a shift register can be set for a particular application by adjusting the number of bits allocated to segment selection and to output stage selection. For example, if 3 bits were allocated to segment selection, instead of 2, up to 8 segments could be utilized instead of just 4. Further, if each segment contained a maximum of 32 or 8 delay

increments, instead of 16, one would allocate 5 or 3 bits, respectively, to output phase selection instead of the 4 bits shown.

[00125] It has been found particularly advantageous to generate all clocks internal to the chip so that they coincide with the rising edge of the chip clock. All noise critical clocks provided external to the chip, are created on the falling edge of the chip clock.

[00126] All clocks in item 140 of Figure 1 are programmable. That is, the division ratio used to obtain a particular clock rate from the chip clock can be programmed. Not only that, they can be programmed during the operation of the chip. The registers setting the dividers for the various clocks can be programmed over the bus using information received over a command line or interface. The arrangement for execution of a change in the programming for a particular clock occurs when a chip sync pulse occurs. This can occur, for example, at a 32 kHz rate.

[00127] Figure 36 shows a plurality of time lines showing clock alignment associated with on-chip generation of clocks in accordance with one aspect of the invention. These time lines illustrate the principles just discussed. In Figure 36, CLK 16 is the clock to which all other clocks are locked. A plurality of additional clocks, CLK 8, CLK 4, CLK 2, CLK 1, CLK 512 and CLK 256 are each derived from CLK 16 by a programmable division, in this case by a power of 2. These clocks operate at 8 MhZ, 4 MhZ, 1 MhZ, 512 KhZ and 256 KhZ, respectively. In addition, an S clock signal is derived and a clock sync signal CLKSYNC occurs every 8 milliseconds which resets the clock dividers and ensures that all clocks operate in lock. A plurality of ADC clocks are shown. These clocks may be, for example, clocks associated with the ADC interface. They are utilized for controlling whatever operations might be desirable within that circuit. In this case, a plurality of different clocks are shown. However, what is important is that each of these clocks utilized with off chip devices are generated on the falling edge of CLK 16. Thus, the activities which occur on the chip shown in Figure 1 will occur at different

instances from the activities occurring on external devices. This provides considerable advantage when dealing with noise and other design issues. The synchronization of clocks on a chip is particularly advantageous because it eases the interfacing of on-chip components because of the known time relationships.

[00128] Figure 37 is a block diagram showing how clock reprogramability is implemented in accordance with the invention. This process is described in conjunction with Figure 36 in which an exemplary 16 megabit per second chip clock is provided to a programmable divider, the divisor of which is received over a bus, (3700) which divides the clock down to a local chip clock frequency. When a new divisor is received over the bus, the clock awaits the arrival of a sync pulse before implementing the change in clock frequency, (3710). A register is connected to the bus so that the value in the register can be programmed from the bus. However, the revised value in the register will not be applied to the programmable divider (3720) until the occurrence of a sync pulse.

[00129] By switching the programming of a clock during the sync pulse, the clock can be reprogrammed during operation without cause causing glitches in the data. Further, data interfacing among devices on the chip is easier when all clocks on the chip are synchronized.

Appendix A shows exemplary compilation tools for obtaining the content of ram1 and ram2 for the Sinc filter stage 2 and for converting bit codes generating from a perl script to DSP code.

[00130] Appendix B is matlab code utilized to generate mini-sinc impulse responses and calculates Sinc filter attenuation and rolloff.

[00131] Appendix C gives programming details for programming the Sinc2 filter.

[00132] Appendix D and Appendix E give exemplary code for the main program and the accumulate and output subroutines for the Sinc filters, respectively.

[00133] Although the present invention has been described and illustrated in detail, it is clearly understood that the same is by way of illustration and example only and is not to be taken by way of limitation, the spirit and scope of the present invention being limited only by the terms of the appended claims and their equivalents.